Ride-sharing in a City

By Aidan Claffey

Shape of the city

- 6x6 grid with a beltway around the city
- Vertical blocks are twice as long as horizontal blocks

My changes to the class model

Speed limits depend on the block:

- Beltway speed limit is 60 mph
- Normal block speed limit is 25 mph

Ubers and normal cars are initialized in the system and never disappear

Normal cars drive around randomly

Ubers pick up and drop off randomly generated passengers (who cannot be picked up or dropped off on the beltway)

Destination is chosen using MATLAB's *graphshortestpath*

How the queue works

Finds the closest Uber with no passenger or queued passenger

If all Ubers are full, it looks for the Uber with destination closest to the location of the passenger (by Euclidean distance) conditional on the Uber having the smallest queue

graphshortestpath()

Parameters are starting vertex, ending vertex, and graph

Function supplies distance and path to take

Path is in vertices, which I turned into a list of blocks instead

graph is where the weights are defined for each block to determine the shortest path

The function uses Dijkstra's algorithm (https://giphy.com/gifs/algorithm-wikimedia-wlQ3KynQ3LX0s/fullscreen)

Treatments (so far)

1. Use distance as the weights
2. Use distance/speed limit as the weights
3. Same as two, but takes into account traffic lights
4. Same as three, but also looks at traffic on each block

Runs

Parameters:
# of ubers: 10

# of non-uber cars: 100

dmin: 20 feet

dmax: 200 feet

Maximum number of passengers: 50

Results (Average passenger wait time)

Traffic optimization model: 0.1530 hours

Speed Limit w/ Traffic Lights: 0.1587 hours

Speed Limit w/o Traffic Lights: 0.1495 hours

Distance: 0.2297 hours

Treatments in the future

- Change beltway to be shorter
- Add a beltway going the other way
- Change uber queue rules to be smarter